UCRL-BR-200562

Building the World's Fastest Linux Cluster

ROBIN GOLDSTONE AND MARK SEAGER

Imagine having 2,304 Xeon processors running day and night solving complex problems. With a theoretical peak of 11.2 teraflops, that is just what the MCR cluster at Lawrence Livermore National Labs (LLNL) is doing. Over the past several years, Lawrence Livermore National Laboratory has deployed a series of increasingly large and powerful Intel-based Linux clusters. The most significant of these is a cluster known as the MCR (Multiprogrammactic Capability Resource). With 1,152 Intel Xeon (2.4 GHz) dual-processor nodes from Linux NetworX and a high performance interconnect from Quadrics, LTD., the MCR currently ranks third on the 21st Top 500 Supercomputer Sites List and is the fastest Linux cluster in the world. This feat was accomplished with a total system cost (hardware including maintenance, interconnect, global file system storage) of under $14 million. Although production clusters like the MCR are still custom built supercomputers that require as much artistry as skill, the experiences of LLNL have helped clear an important path for other clusters to follow.

Building on Experience

Livermore has been at the forefront of HPC for many years. Livermore Computing (LC) provides the majority of unclassified and classified computing cycles to scientific programs throughout Livermore. Recent successes with Linux clusters mostly built from COTS (Commodity-Off-The-Shelf) components indicate that a large portion of Livermore's computing requirements can be served with these cost-effective, stable, and easily manageable systems.

LC clusters fall into two architectural categories: capability clusters, which can run large-scale, distributed-memory applications that consume at least 50% of the nodes, and capacity clusters, which are either a loosely coupled collection of nodes that run a serial workload or tightly coupled clusters that run small and medium scale distributed-memory applications.

All cluster designs adhere to the the Livermore Model, which abstracts the parallelism and I/O services of the cluster computing environment to a high level. The abstraction is based on SMP compute nodes attached to a high-speed, low-latency message passing interconnects. Applications utilize compiler-generated OpenMP threads to exploit shared memory parallelism and message passing interface (MPI) to exploit distributed memory parallelism between nodes. Data sets are stored on a common (POSIX interface) parallel file system, which may utilize striping across multiple file service nodes in order to achieve the required bandwidth and scalability.

Both capability and capacity clusters share a common cluster architecture that includes one or two system management nodes, a private management IP network, a serial console/remote power management infrastructure, a public IP network that makes nodes externally visible, and one or more dedicated login nodes. Capability systems, like the MCR, have additional common architectural traits required to implement the Livermore Model, including high-speed, message passing interconnect and a storage infrastructure for parallel I/O.

The MCR cluster includes management nodes provide dedicated resources for system administration of the cluster including OS installation and software updates, remote power/console management, system monitoring, and resource management. There are several designated login nodes that serve as the interactive access point for the end-users of the cluster. Login nodes are a place for development and compilation of applications, batch job submission, workload monitoring, review of results and offload of data to the LLNL archive or other systems for post-processing.

A Quadrics QsNet high-speed interconnect is used for message passing, for bulk data movement to and from a parallel file system, and to carry IP traffic for the public network. The interconnect is normally attached to all nodes except the management nodes. Login nodes typically act as IP gateways, routing traffic between the high-speed interconnect and the Livermore backbone networks.

Livermore is focusing significant effort on the development and deployment of a parallel file system for Linux. These efforts are focused on Lustre, which is rap-

idly approaching production-readiness on large clusters such as MCR. Two storage architecture approaches are currently in use with Lustre. Both approaches use some number of dedicated I/O nodes over which the rest of the cluster stripes data transfers to achieve the required parallel I/O bandwidth. In one approach, Lustre Object Storage Targets (OSTs) are directly attached to the I/O nodes via Fibre Channel. In the other, the I/O nodes act as gateways for network-attached OSTs. In the Lustre model, several cluster nodes are also designated as MetaData Servers (MDSs). On clusters where Lustre is not yet deployed, I/O gateway nodes may provide access to high-performance NFS servers on a private network

Choices, Choices, Choices

For every large production cluster procured, a small test cluster of the same architecture is deployed in a testbed where it is used to test new software and system configurations before they are tried on the production platforms. Vendors are requested to ship the test cluster to Livermore as early as possible so that LC can work in parallel with them toward solving the inevitable problems that will delay acceptance.

When building Linux clusters with COTS components, it is important to recognize that specifying components require a much greater level of care than than those required to build a desktop PC. Motherboard chip sets need kernel support for environmental sensor access and memory error detection. Hardware incompatibilities may arise between the BIOS and high-performance peripheral adapters. Hardware that is well supported under other operating systems might not perform as well, be as stable, or work at all under Linux.

Subtle differences in hardware can manifest themselves as significant differences in the overall manageability and mean time between failure (MTBF) of a large cluster. It is therefore important to be clear and detailed with regard to the functionality, performance and quality required of new hardware and to test as much as possible in advance before committing to a large amount of hardware.

Fast processors are throttled by a slow interconnect. For this reason, LC exclusively uses the Quadrics QsNet Elan3 interconnect for the MCR. Elan3 delivers 4.5 µs MPI latency and 320 MB/s exchange (MPI_SENDREC) bandwidth between user processes on a pair of nodes. Results vary depending on motherboard chip set so it is important establish minimum interconnect performance requirements in conjunction with node requirements.

Remote manageability is a practical necessity for large clusters and includes serial port access and remote power control. Node serial ports are connected to terminal servers that support "reverse Telnet," in which each serial port on the server is directly accessed through a unique IP address:port pair. LC console management software maintains a persistent connection to each serial port for continuous logging and interactive access. Remote control of power requires switchable plugs that can be controlled via telnet over a serial port or network interface. LC power control software uses an Expect-like scripting language to control the plugs on these devices which provide "plug on," "plug off," and "plug status" commands. On large clusters, devices that control multiple plugs are preferred over devices that control only one (such as an embedded node service processor with its own serial port or Ethernet connection), because this reduces the amount of terminal server/Ethernet switch infrastructure and cabling required. It also keeps down the number of sockets that need to be held open by the power control software. Remote power units and terminal servers are typically attached to a network that is only reachable from nodes with restricted user access.

Putting It Together

It may seem that building a commodity Linux cluster is a simple matter of stacking a bunch of PCs in a rack. Yet, this seemingly straightforward task is not simple. Many minor details need attention, and it can require a lot of time to assemble a collection of parts into a large cluster. While LC system administrators have successfully built and deployed clusters of up to 70 nodes without an integrator, the process was tedious and not very cost effective. Unless inexpensive labor is available, anything larger than a single rack cluster is probably worthy of some amount of vendor integration services.

The cluster will need racks, and all racks are not created equally. A high-quality rack with flexible mounting options and generous space for cable routing will improve the quality of the finished product. Wherever possible, network switches, power controllers, and terminal servers should be located to minimize cable length and cable crossover between racks. Heavier components should be placed lower in the racks for stability and increased ease of service. For aesthetic and airflow purposes, blank spaces should be consolidated at the tops of racks, and blank filler plates should be used. Power switches should be covered by rack doors or otherwise located or protected so that they won't be accidentally turned off.

A rack full of tangled wires is an eyesore and a serviceability nightmare. Cable lengths should be opti-

mized to avoid excessive coils of unneeded cables, and cables should be routed down the sides of the racks and tied off with Velcro bands. Plastic zip ties should be avoided, because they make it difficult to reroute or replace cables. Cables should not be crimped or bent at excessive angles. DB-type cables should be screwed down to ensure good connections. RJ-type cables should use non-hooded connectors, because rubber hoods can impede the ability to make a secure connection and make it difficult to release in tight spaces such as dense Ethernet switches.

All cables in the cluster should be labeled at both ends with a source and destination identifier. This will aid in the debugging of any connection problems and ensure that components can be removed and replaced reliably.

A New Hardware Support Strategy

Typical COTS components such as motherboards, CPUs, memory, and hard drives are relatively inexpensive and readily available. Therefore, instead of a more tradition support contract, COTS clusters allow a do-it-yourself hardware maintenance that can be a cost-effective and reliable strategy, even for production installations with high availability requirements.

LC has adopted a support model that makes use of a "cold" parts cache, a "hot spare" cluster, and appropriately skilled staff members. When a new cluster is purchased, a number of additional components are procured and stored on a shelf. The quantity of each component is based on published MTBF data as well as locally collected data on similar hardware. Other factors in sizing the parts cache include the manufacturer's warranty for each component and the tolerance level for substitution of non-identical components. A hot spare cluster is used to troubleshoot nodes with non-obvious hardware problems and also to burn in new hardware before deployment into production clusters.

CHAOS on Clusters

LC's high performance Linux cluster environment brings together a unique set of hardware and software requirements at a very large scale, which precludes the use of a standard "off the shelf" Linux software distribution. As a result, LC began producing the Clustered High Availability Operating System (CHAOS), its in-house Linux distribution for HPC clusters. CHAOS is derived from a standard Linux distribution, currently the Red Hat 7.3 boxed set with plans to transition to Red Hat Enterprise Linux 3.0 in the near future. The standard distribution is first customized by removing unneeded Red Hat packages based on user and system requirements. We then add a number of HPC enhancements including a modified CHAOS kernel (see sidebar, pg. xx), cluster management tools, resource management, support for the Lustre parallel file system, authentication and access control and tools for monitoring and failure detection. The CHAOS strategy is to leverage our vendor partnership (Red Hat) for the majority of software components that are not Livermore or cluster specific.

The CHAOS kernel actively tracks Red Hat errata kernel releases with modifications needed to support unique LC hardware (e.g. Quadrics interconnect) and non-mainstream software technologies (e.g. Lustre). Kernel crash dump support and serial console logging enhance LC's ability to support the Linux kernel.

The Red Hat kernel series was chosen because the kernel is compatible with other Red Hat components, allows LC to leverage a support relationship with Red Hat kernel developers, and provides a kernel somewhat more closely aligned to Livermore's needs than the stable Linux kernel releases.

When a kernel is ready for release, it is tagged and built as an RPM then subjected to a variety of regression tests before being deemed suitable for installation on LC's production systems.

A good set of cluster system management tools is essential to the successful operation of large clusters. The CHAOS cluster system management tool set (see sidebar, pg. xx), has been successful in meeting the needs of LC Linux system administrators and has enabled systems the size of MCR to be managed with less than one full-time system administrator.

Application Development Environment

Compilers
- Intel compilers
- GNU compiler
- PGI compilers

MPI
- Quadrics MPI (MPICH based)

Debugging
- TotalView

Hardware counter tool
- PAPI (open source)

Memory correctness tool
- Valgrind (open source)

Parallel profiling tool
- KAI (Intel)VGV
- Pallas (Intel) Vampirtrace

Intel Math Kernel Library
- MKL

TABLE ONE

CHAOS Kernel Patches

PATCH	DESCRIPTION
QsNet Elan	Quadrics Elan3 adapter and related drivers (added)
Qlogic qla2300	Qlogic 2 gigabit FibreChannel adapter (updated)
MTD ich2rom	Intel firmware hub flash device (added)
E1000	Intel Eepro1000 GigE adapter (updated)
AceNIC	Netgear GA620 GigE adapter (updated)
MCORE/BOOTIMG	Mission Critical Linux Crash Dump
QsNet	Adapter MMU sync, exit handlers, core file naming
TV ptrace	Ptrace changes for TotalView parallel debugger
PerfCtr	Hardware performance counter support
ECC	Poll memory errors and optionally panic the node
p4therm	Detect Xeon thermal throttling/optionally panic the node
Lustre support	VFS intents, RO devices, zero-copy TCP enhancement
statfs64	Support for statfs64() system call
NFS groups > 16	Membership in > 16 groups on NFS file systems
NFS zombie	Bug fix for un-killable zombies blocked in NFS I/O
Dsp stack trace	Use frame pointers to improve oops stack traces
FDs > 1024	Allow 8192 file descriptors per process

As described previously, the Livermore Model seeks to provide a common application programming framework across all LC production systems. Different operating systems and architectures may constrain this goal, in which case the focus shifts to providing the required functionality even if the actual tools differ from platform to platform. A list of standard LC application development tools is provided in the sidebar.

Monitoring and Failure Detection

Detecting hardware and software failures and acting on them quickly is crucial to ensuring that a cluster functions properly and delivers reliable service to customers. LC clusters employ system monitoring, kernel level fault detection, and boot-time checks to detect failures that may prevent effective use of these systems.

On a cluster running computations where the answers matter, an uncorrectable memory error should result in a kernel panic. The CHAOS kernel has been modified to detect memory ECC errors using the ECC module for chip sets that LC has deployed. A /proc file allows SNMP to obtain and report correctable error counts by memory module, which, if excessive, indicate that memory should be proactively replaced. When an uncorrectable error occurs, the kernel panics, calling out the offending memory module. This effectively removes the node from play and terminates any running application that may have been affected by bad data.

Resource Management and the Lustre FS

LC provides a common batch scheduling and resource accounting system across all its platforms called DPCS (Distributed Production Control System). It is a "meta-batch" system in the sense that users can request execution on the first cluster that has resources available that meet a list of constraints. On Linux clusters with the Quadrics interconnect, DPCS interfaces with the SLURM resource manager, an open source replacement for Quadrics' RMS product.

A parallel file system is required on capability systems that implement the Livermore Model. The file system must provide a POSIX interface, perform well, and be extremely stable. As mentioned previously, LC's Linux parallel file system strategy is based on Lustre. Lustre uses OSTs (object storage targets) to implement its storage protocol. OSTs may be implemented as RAID devices on a Fibre Channel SAN, NAS devices or Linux cluster nodes with locally-attached storage such as SATA RAID. The first phase of Lustre, known as Lustre Lite, has achieved performance and stability targets and is slated to be used in production on the MCR clus-

ter in the fall of 2003.

Lessons Learned

The deployment of LC Linux clusters has been a huge success, though we have learned a number of lessons from mistakes along the way.

Lesson 1: Caveat emptor. It is important to choose a cluster provider based on a best-value procurement style, not on the basis of lowest bid. Use of poor quality components or the lack of attention to configuration details can lead to unstable systems. Considerable attention needs to be paid to every aspect and assumption of the system configuration.

Lesson 2: Expect the unexpected. The integration process must be flexible enough to allow for dealing with problems as (not if) they arise. For example, during the initial testing of LC's first production HPC Linux cluster, some nodes produced wrong answers. After much experimentation and testing, the primary cause of the problem determined to be uncorrectable memory errors. On some motherboards, Linux ignores these errors and continues.

Lesson 3: Performance problems are a bug. The test and/or acceptance phase of the integration process must include the following components: 1) check to see that everything is there; 2) check to see if critical aspects of system performance meet contractual requirements; 3) check to see if the system is stable over a reasonably long period of time (one or two weeks). For example, when MCR was being integrated, jobs were unexpectedly slow some of the time. It was found that the Xeon processors were dropping to a reduced clock duty cycle due to thermal throttling. Following this discovery, the p4therm kernel module was developed and deployed and hardware configuration and facilities modifications were performed to really solve the heat problems.

Lesson 4: Large clusters require fully automatic management functions. It is also important to ensure that Linux-based tools are available for upgrading BIOSes and changing CMOS settings for the nodes. In the past, on another cluster, integration personnel actually used DOS boot floppies to flash the BIOSes on almost one thousand nodes. Tools to automate this and other processes are now part of the CHAOS software stack.

Future Challenges

LC faces a number of challenges in order to build the next generation of clusters and expand our overall simulation environment. The Lustre file system will be a critical technology component for this transition.

In addition, the programmatic mission at Livermore is driving us to build even larger clusters. There are now several viable choices for cost effective nodes based on two-way and four-way SMPs with 64 bit CPUs. We are in the process of procuring a 1024 node cluster of four-way Itanium nodes with over 22 teraFLOP/s peak capability. Exploiting 64-bit Linux as well as technologies such as IPMI will present interesting challenges.

With the advent of Infiniband, high performance networking technology is now available from multiple sources. It appears that the commoditization of high performance low latency networks is under way.

On the I/O front, Serial ATA (SATA), Serial SCSI and low cost high-performance RAID5 adapters should further commoditize the NAS marketplace and lead to commodity Lustre OSTs from multiple sources.

The MCR will eventually lose its stature on the Top

Cluster System Management Tools

YACI (Yet Another Cluster Installer) builds node images in a chrooted environment, boots a standalone OS over the network on each node, copies images to the nodes using either an NFS pull or a multicast push and then installs the local disk.

Pdsh (Parallel Distributed Shell) is a high-performance, parallel remote shell utility.

ConMan is a serial console management program designed to support a large number of console devices and simultaneous users.

PowerMan is a tool for controlling multiple remote power control (RPC) devices in parallel via a command-line interface.

Genders is a collection of utilities used as a basis for cluster configuration management.

Croute is a Perl script that sets static routes on the basis of the IP addresses of locally configured interfaces and information in a config file. Large clusters with multiple networks often need complex network routing schemes to achieve load balancing of traffic across multiple interfaces.

Munge is a scalable intra-cluster authentication protocol allowing a remote process to securely authenticate the UID/GID of the originating process.

SLURM is an open source replacement for Quadrics' RMS resource manager product.

See *Resources* for download information.

500 list. However, the trail it has blazed for the next generation of clusters will certainly be remembered for years to come.

Resources

MCR Page
- www.llnl.gov/linux/mcr/mcr.html

Linux NetworX
- www.lnxi.com

Quadrics
- www.quadrics.com

Top500 List
- www.top500.org/list/2003/06/

CHAOS Kernel
- www.llnl.gov/linux/chaos/chaos.html
- www.anime.net/~goemon/linux-ecc/

Cluster System Management Tools
- www.llnl.gov/linux/downloads.html
- www.llnl.gov/linux/slurm/

Intel Tools
- www.intel.com/software/products/

Portland Group
- www.pgi.com

TotalView debugger
- www.etnus.com

PAPI Hardware counter tool
- icl.cs.utk.edu/papi

Valgrind Memory correctness tool
- developer.kde.org/~sewardj

DPCS
- www.llnl.gov/icc/lc/dpcs/dpcs_overview.html

Lustre File System
- www.lustre.org/